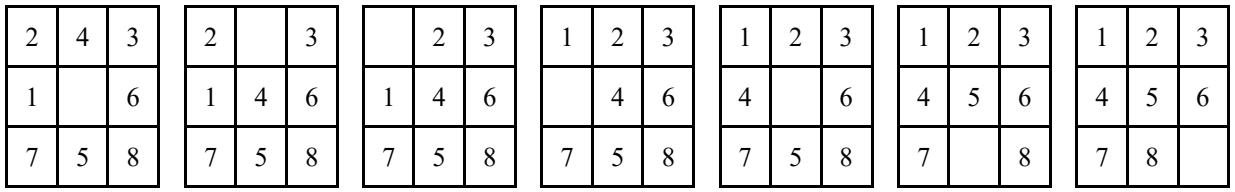
11. Svetlana

Program Name: Svetlana.java **Input File: svetlana.dat**

Svetlana's favorite toy growing up was the 8 puzzle game. For those unfamiliar with the 8 puzzle game, the goal of the 8 puzzle game is to get a random ordering of the numbers 1 thru 8, into acceding order, leaving a blank at the bottom, right most corner. The only legal move is to slide a number into a blank. No two numbers, can be swapped. For example, take the below board. This board would take 6 moves to optimally get all blocks in the correct ascending order 1 thru 8, with the blank at the bottom right most position.



Move 1	Move 2	Move 3	Move 4	Move 5	Move 6
Slide 4 into BLANK	Slide 2 into BLANK	Slide 1 into BLANK	Slide 4 into BLANK	Slide 5 into BLANK	Slide 8 into BLANK

Although Svetlana, can solve this puzzle, she wants to know if she's solving it in the best, optimal way. Can you help Svetlana write a program that given a starting board, will output the number of moves it would take to optimally solve the puzzle?

Input: Input will begin with an integer N, the number of test cases. N will be in range [1,10]. Each starting board is preceded by eight dashes. This serves to break up each test case, making it easier to read. Following the eight dashes, will be the 3 x 3 board, representing the starting position of the tiles. The value of `-1` will be used to indicate the BLANK position in the board. It is guaranteed that the numbers 1-8 will be given. There will be no duplicates, and no numbers outside of the range [1,8].

Output: For each test case, you are to output: "`Number of steps needed to solve: #`", where # is the number of moves to optimally solve the puzzle. If the puzzle has no solution, output: "`No solution exists.`"

~ ***Sample input and output are on next page…*** ~

~ *Svetlana continued…* ~

Sample input:

```
7
--------
2 4 3
1 -1 6
7 5 8
--------
1 2 3
4 5 6
7 8 -1
--------
1 2 3
8 -1 4
7 6 5
--------
1 2 3
4 5 6
7 -1 8
--------
2 7 3
8 1 5
4 6 -1
--------
2 1 -1
3 6 5
7 8 4
--------
8 6 7
2 5 4
3 -1 1
```

Sample output:

```
Number of steps needed to solve: 6
Number of steps needed to solve: 0
No solution exists.
Number of steps needed to solve: 1
Number of steps needed to solve: 14
Number of steps needed to solve: 20
Number of steps needed to solve: 31
```